完整的代码文档：

### **InstrumentBase类（通用仪表基类）**

1. **\_\_init\_\_**方法****：

* 继承QWidget类的初始化过程，完成基本的窗口部件初始化。
* 规定仪表的最小尺寸为 150x150 像素，确保仪表在界面上有合适的显示大小，避免因过小而导致显示不清或布局混乱。
* 通过setStyleSheet应用 QSS 样式，将背景颜色设置为深灰色（#222），赋予界面一种沉稳、工业风的基调。设置边框半径为 10 像素，使仪表边缘呈现出圆润的效果，增加美观度。边框宽度为 2 像素，颜色为稍浅的灰色（#444），突出显示每个仪表，增强其视觉独立性和层次感。

1. **paintEvent方法：**

* 此方法是QWidget类用于绘制界面内容的关键方法。在该方法中创建QPainter对象，QPainter是用于在QWidget上进行绘制操作的类。
* 调用draw\_background和draw\_indicator方法，这两个方法在具体的仪表子类中实现。这种设计模式遵循了面向对象编程中的多态性原则，允许不同类型的仪表（子类）以各自独特的方式绘制背景和指示器，同时保持基类方法的统一调用结构。

### **Gauge类（仪表盘）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承父类的属性和初始化设置，包括样式和最小尺寸等。
* 初始化仪表盘特有的属性：当前值value（初始化为 0）、最小值min\_value（0）、最大值max\_value（100）和标签label（'Value'），这些属性定义了仪表盘的测量范围和标识信息。
* 创建一个QTimer定时器对象，将其timeout信号连接到update\_value方法。定时器每 1000 毫秒（1 秒）触发一次timeout信号，用于周期性地更新仪表盘的值，实现动态显示效果。

1. **update\_value方法：**

* 实现仪表盘值的更新逻辑。每次调用时，将当前值value增加 1，然后通过取模操作（% (self.max\_value + 1)）确保值始终在min\_value到max\_value所定义的范围内循环变化。例如，当value达到max\_value后，下一次更新会使其回到min\_value重新开始计数。
* 调用update方法，此方法继承自QWidget类。它会触发paintEvent方法的调用，从而导致仪表界面重绘，显示更新后的数值。

1. **draw\_background方法：**

* 首先获取仪表所在矩形区域的中心坐标center，通过计算仪表宽度和高度的较小值的一半并乘以 0.8 得到半径radius，以此来确定仪表盘的圆形范围。然后使用drawEllipse方法绘制仪表盘的圆形轮廓，该方法基于给定的中心坐标和半径在QPainter对象上绘制椭圆（在这里是圆形）。
* 设置绘制文本的字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环遍历min\_value到max\_value（步长为 10）的每个刻度值，根据value\_to\_angle方法计算每个刻度值在圆形仪表盘上对应的角度，进而计算出刻度值在圆形上的坐标位置，最后使用drawText方法将刻度值绘制在仪表盘上。

1. **value\_to\_angle方法：**

* 根据给定的值计算其在仪表盘圆周上对应的角度。通过将值与最小值的差值除以最大值与最小值的差值，得到一个比例值，再将该比例值乘以 2π（整个圆周对应的弧度值），最后减去 π/2，将角度范围映射到从 -π/2 到 3π/2，以适应仪表盘的指针旋转方向和刻度布局。

1. **draw\_indicator方法：**

* 根据当前值value计算指针的角度和长度。通过value\_to\_angle方法获取当前值对应的角度，然后根据半径的一定比例（0.6）计算指针长度。基于仪表中心坐标、指针角度和长度，使用drawLine方法绘制指针，颜色设为红色，宽度为 3 像素，使指针在仪表盘中清晰可见。
* 设置用于显示当前值和标签的字体为加粗的Arial字体，字号为 12，颜色为白色。在仪表盘中心下方合适的位置，使用drawText方法显示当前值和标签，让用户清晰地读取仪表所表示的信息。

### **Thermometer类（温度计）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承通用的仪表属性和样式。
* 初始化温度计特有的属性：当前温度值value（20）、最小值min\_value（0）、最大值max\_value（100）和温度单位unit（'°C'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 2000 毫秒（2 秒）触发一次，用于定期更新温度值，实现温度计的动态变化效果。

1. **update\_value方法：**

* 更新温度值。每次调用将当前温度值value增加 5，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，以模拟温度的变化情况。
* 调用update方法触发重绘，使温度计显示新的温度值。

1. **draw\_background方法：**

* 使用drawRect方法绘制温度计的外壳，指定矩形的左上角坐标（30, 50）和宽度、高度（20, 200），创建一个简单的矩形表示温度计的形状。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据温度范围（步长为 10）计算每个刻度值对应的y坐标，将刻度值绘制在温度计旁边，为用户提供温度刻度参考。

1. **draw\_indicator方法：**

* 根据当前温度值计算温度柱的y坐标，使用drawRect方法绘制红色的温度柱。温度柱的高度根据温度值在min\_value到max\_value范围内的比例来确定，通过计算得到温度柱在温度计外壳矩形内的起始y坐标和高度，从而实现温度的可视化显示。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在温度柱顶部附近合适的位置，使用drawText方法显示当前温度值和单位，方便用户读取温度信息。

### **Manometer类（压力计）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，获取通用的仪表设置。
* 初始化压力计的压力值value（1.0）、最小值min\_value（0）、最大值max\_value（2.0）和压力单位unit（'MPa'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 1500 毫秒（1.5 秒）触发一次，用于更新压力值，实现压力计的动态显示。

1. **update\_value方法：**

* 更新压力值。将当前压力值value增加 0.1，使用round函数四舍五入后，通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟压力的变化情况。
* 调用update方法触发重绘，使压力计显示新的压力值。

1. **draw\_background方法：**

* 获取仪表中心坐标，使用drawArc方法绘制压力计的弧形轮廓。指定弧形的外接矩形左上角坐标（相对于中心坐标）、宽度和高度（140, 140），以及弧形的起始角度（45 \* 16，角度乘以 16 是QPainter的绘制要求）和跨度角度（270 \* 16），创建出压力计的弧形背景。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据压力值范围（步长为 0.1）计算每个刻度值对应的角度，将刻度值绘制在压力计上，为用户提供压力刻度参考。

1. **value\_to\_angle方法：**

* 根据给定的压力值计算其在压力计弧形上对应的角度。通过将压力值与最小值的差值除以最大值与最小值的差值，得到一个比例值，再将该比例值乘以 2.5π（根据弧形的角度范围设定），最后减去 π/4，将角度范围映射到从 -π/4 到 5π/4，以适应压力计的弧形刻度布局。

1. **draw\_indicator方法：**

* 根据当前压力值计算指针的角度和长度，使用drawLine方法绘制指针，颜色设为蓝色，宽度为 3 像素。指针的位置根据当前压力值对应的角度和指定长度计算得出，使其准确地指示压力值在弧形刻度上的位置。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在压力计下方合适的位置，使用drawText方法显示当前压力值和标签，便于用户读取压力信息。

### **Voltmeter类（电压表）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承通用仪表的属性和样式。
* 初始化电压表的电压值value（220）、最小值min\_value（0）、最大值max\_value（380）和电压单位unit（'V'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 3000 毫秒（3 秒）触发一次，用于更新电压值，实现电压表的动态显示效果。

1. **update\_value方法：**

* 更新电压值。将当前电压值value增加 10，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟电压的变化情况。
* 调用update方法触发重绘，使电压表显示新的电压值。

1. **draw\_background方法：**

* 使用drawRect方法绘制电压表的外壳，指定矩形的左上角坐标（30, 30）和宽度、高度（100, 200），创建一个表示电压表外观的矩形。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据电压范围（步长为 20）计算每个刻度值对应的y坐标，将刻度值绘制在电压表旁边，为用户提供电压刻度参考。

1. **draw\_indicator方法：**

* 根据当前电压值计算指示条的y坐标，使用drawRect方法绘制绿色的指示条。指示条的高度根据电压值在min\_value到max\_value范围内的比例确定，通过计算得到指示条在电压表外壳矩形内的起始y坐标和高度，实现电压值的可视化显示。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在指示条顶部附近合适的位置，使用drawText方法显示当前电压值和单位，方便用户读取电压信息。

### **Ammeter类（电流表）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承通用仪表的初始化设置。
* 初始化电流表的电流值value（5）、最小值min\_value（0）、最大值max\_value（10）和电流单位unit（'A'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 2500 毫秒（2.5 秒）触发一次，用于更新电流值，实现电流表的动态显示。

1. **update\_value方法：**

* 更新电流值。将当前电流值value增加 1，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟电流的变化情况。
* 调用update方法触发重绘，使电流表显示新的电流值。

1. **draw\_background方法：**

* 使用drawRect方法绘制电流表的外壳，指定矩形的左上角坐标（30, 30）和宽度、高度（100, 200），创建电流表的外观矩形。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据电流范围（步长为 1）计算每个刻度值对应的y坐标，将刻度值绘制在电流表旁边，为用户提供电流刻度参考。

1. **draw\_indicator方法：**

* 根据当前电流值计算指示条的y坐标，使用drawRect方法绘制黄色的指示条。指示条的高度根据电流值在min\_value到max\_value范围内的比例确定，通过计算得到指示条在电流表外壳矩形内的起始y坐标和高度，实现电流值的可视化显示。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在指示条顶部附近合适的位置，使用drawText方法显示当前电流值和单位，方便用户读取电流信息。

### **LevelMeter类（液位计）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，获取通用仪表的属性和样式。
* 初始化液位计的液位值value（50）、最小值min\_value（0）、最大值max\_value（100）和液位单位unit（'%'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 2000 毫秒（2 秒）触发一次，用于更新液位值，实现液位计的动态显示效果。

1. **update\_value方法：**

* 更新液位值。将当前液位值value增加 5，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟液位的变化情况。
* 调用update方法触发重绘，使液位计显示新的液位值。

1. **draw\_background方法：**

* 使用drawRect方法绘制液位计的外壳，指定矩形的左上角坐标（30, 30）和宽度、高度（100, 200），创建液位计的外观矩形。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据液位范围（步长为 10）计算每个刻度值对应的y坐标，将刻度值绘制在液位计旁边，为用户提供液位刻度参考。

1. **draw\_indicator方法：**

* 根据当前液位值计算指示条的y坐标，使用drawRect方法绘制橙色的指示条。指示条的高度根据液位值在min\_value到max\_value范围内的比例确定，通过计算得到指示条在液位计外壳矩形内的起始y坐标和高度，实现液位值的可视化显示。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在指示条顶部附近合适的位置，使用drawText方法显示当前液位值和单位，方便用户读取液位信息。

### **FlowMeter类（流量计）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承通用仪表的属性和样式。
* 初始化流量计的流量值value（100）、最小值min\_value（0）、最大值max\_value（200）和流量单位unit（'L/min'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 1000 毫秒（1 秒）触发一次，用于更新流量值，实现流量计的动态显示效果。

1. **update\_value方法：**

* 更新流量值。将当前流量值value增加 10，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟流量的变化情况。
* 调用update方法触发重绘，使流量计显示新的流量值。

1. **draw\_background方法：**

* 获取仪表中心坐标，使用drawEllipse方法绘制流量计的圆形轮廓，指定半径为 60，以中心坐标为圆心绘制圆形。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据流量范围（步长为 20）计算每个刻度值对应的角度，将刻度值绘制在流量计上，为用户提供流量刻度参考。这里使用value\_to\_angle方法将流量值转换为对应的角度，与圆形流量计的刻度布局相匹配。

1. **value\_to\_angle方法：**

* 根据给定的流量值计算其在流量计圆周上对应的角度。通过将流量值与最小值的差值除以最大值与最小值的差值，得到一个比例值，再将该比例值乘以 2π（整个圆周对应的弧度值），最后减去 π/2，将角度范围映射到从 -π/2 到 3π/2，以适应流量计的圆形刻度布局和指针旋转方向。

1. **draw\_indicator方法：**

* 根据当前流量值计算指针的角度和长度。首先通过value\_to\_angle方法获取当前流量值对应的角度，然后根据圆形半径的一定比例确定指针长度。使用drawLine方法绘制指针，颜色设为洋红色，宽度为 3 像素，使指针在流量计圆形轮廓上清晰可见。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在流量计下方合适的位置，使用drawText方法显示当前流量值和标签，方便用户读取流量信息。

### **Tachometer类（转速表）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承通用仪表的属性和样式。
* 初始化转速表的转速值value（1000）、最小值min\_value（0）、最大值max\_value（2000）和转速单位unit（'RPM'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 1500 毫秒（1.5 秒）触发一次，用于更新转速值，实现转速表的动态显示效果。

1. **update\_value方法：**

* 更新转速值。将当前转速值value增加 100，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟转速的变化情况。
* 调用update方法触发重绘，使转速表显示新的转速值。

1. **draw\_background方法：**

* 获取仪表中心坐标，使用drawEllipse方法绘制转速表的圆形轮廓，大小经过调整。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据转速范围（步长为 200）计算每个刻度值对应的角度，将刻度值绘制在转速表上，为用户提供转速刻度参考。

1. **value\_to\_angle方法：**

* 根据给定的转速值计算其在转速表圆周上对应的角度。通过将转速值与最小值的差值除以最大值与最小值的差值，得到一个比例值，再将该比例值乘以 2π（整个圆周对应的弧度值），最后减去 π/2，将角度范围映射到从 -π/2 到 3π/2，以适应转速表的圆形刻度布局和指针旋转方向。

1. **draw\_indicator方法：**

* 根据当前转速值计算指针的角度和长度，使用drawLine方法绘制指针，颜色设为青色，宽度为 3 像素。指针的位置根据当前转速值对应的角度和指定长度计算得出，使其准确地指示转速值在圆形刻度上的位置。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在转速表下方合适的位置，使用drawText方法显示当前转速值和单位，便于用户读取转速信息。

### **Hygrometer类（湿度计）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承通用仪表的属性和样式。
* 初始化湿度计的湿度值value（50）、最小值min\_value（0）、最大值max\_value（100）和湿度单位unit（'%RH'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 2500 毫秒（2.5 秒）触发一次，用于更新湿度值，实现湿度计的动态显示效果。

1. **update\_value方法：**

* 更新湿度值。将当前湿度值value增加 5，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟湿度的变化情况。
* 调用update方法触发重绘，使湿度计显示新的湿度值。

1. **draw\_background方法：**

* 使用drawRect方法绘制湿度计的外壳，位置和尺寸经过调整。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据湿度范围（步长为 10）计算每个刻度值对应的y坐标，将刻度值绘制在湿度计旁边，为用户提供湿度刻度参考。

1. **draw\_indicator方法：**

* 根据当前湿度值计算指示条的y坐标，使用drawRect方法绘制紫色的指示条。指示条的高度根据湿度值在min\_value到max\_value范围内的比例确定，通过计算得到指示条在湿度计外壳矩形内的起始y坐标和高度，实现湿度值的可视化显示。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在指示条顶部附近合适的位置，使用drawText方法显示当前湿度值和单位，方便用户读取湿度信息。

### **11. PowerMeter类（功率计）**

1. **\_\_init\_\_方法：**

* 调用父类InstrumentBase的\_\_init\_\_方法，继承通用仪表的属性和样式。
* 初始化功率计的功率值value（1000）、最小值min\_value（0）、最大值max\_value（2000）和功率单位unit（'W'）。
* 创建一个QTimer定时器，连接到update\_value方法，定时器每 2000 毫秒（2 秒）触发一次，用于更新功率值，实现功率计的动态显示效果。

1. **update\_value方法：**

* 更新功率值。将当前功率值value增加 100，然后通过取模操作确保值在min\_value到max\_value范围内循环变化，模拟功率的变化情况。
* 调用update方法触发重绘，使功率计显示新的功率值。

1. **draw\_background方法：**

* 获取仪表中心坐标，使用drawEllipse方法绘制功率计的圆形轮廓，大小经过调整。
* 设置字体为加粗的Arial字体，字号为 10，颜色为浅灰色。通过循环根据功率范围（步长为 200）计算每个刻度值对应的角度，将刻度值绘制在功率计上，为用户提供功率刻度参考。

1. **value\_to\_angle方法：**

* 根据给定的功率值计算其在功率计圆周上对应的角度。通过将功率值与最小值的差值除以最大值与最小值的差值，得到一个比例值，再将该比例值乘以 2π（整个圆周对应的弧度值），最后减去 π/2，将角度范围映射到从 -π/2 到 3π/2，以适应功率计的圆形刻度布局和指针旋转方向。

1. **draw\_indicator方法：**

* 根据当前功率值计算指针的角度和长度，使用drawLine方法绘制指针，颜色设为深黄色，宽度为 3 像素。指针的位置根据当前功率值对应的角度和指定长度计算得出，使其准确地指示功率值在圆形刻度上的位置。
* 设置字体为加粗的Arial字体，字号为 12，颜色为白色。在功率计下方合适的位置，使用drawText方法显示当前功率值和标签，便于用户读取功率信息。

### **主程序部分**

在主程序中，首先创建了一个QApplication实例app，这是 Qt 应用程序的入口点，负责管理应用程序的生命周期和事件循环。

创建一个QGridLayout布局对象layout，网格布局可以方便地将多个仪表以规则的网格形式排列在界面上。

创建十个不同类型的仪表实例，并将它们添加到列表widgets中。每个仪表实例都具有独特的属性和动态更新功能，通过定时器不断更新其显示的值。

通过循环将每个仪表添加到layout布局中，每添加一个仪表，列索引col增加 1。当列索引达到 5 时，换行并将列索引重置为 0，行索引row增加 1。这样可以确保仪表以 5 列的网格形式排列在界面上。

创建主窗口main\_widget，将布局layout设置给主窗口，使布局中的仪表能够在主窗口中显示。然后调用show方法显示主窗口，让用户能够看到仪表界面。

最后，通过调用app.exec\_()启动应用程序的事件循环。在事件循环中，应用程序会不断处理各种事件，如用户输入、定时器触发等。当用户关闭窗口时，程序退出事件循环，结束应用程序的运行。

### **1. InstrumentBase Class (General Instrument Base Class)**

#### **\_\_init\_\_ Method**

* This method initializes the InstrumentBase class which inherits from QWidget. It sets up the basic properties of the instrument widget.
* A minimum size of 150x150 pixels is set for the widget. This ensures that the instrument has an appropriate display area and doesn't appear too small or cause layout issues.
* The style sheet is applied using setStyleSheet. The background color is set to #222, giving it a dark and industrial look. The border - radius is 10 pixels, making the edges rounded, and the border is 2 pixels wide with a color of #444, which gives a distinct and professional appearance to the instrument.

#### **paintEvent Method**

* When the widget needs to be painted, this method is called. It creates a QPainter object which is used for drawing operations on the widget.
* It calls the draw\_background and draw\_indicator methods. These methods are meant to be overridden in the derived classes to provide specific drawing functionality for different types of instruments. This design follows the principle of polymorphism, allowing for customized drawing of each instrument type while maintaining a common structure in the base class.

### **2. Gauge Class (Dashboard)**

#### **\_\_init\_\_ Method**

* Calls the \_\_init\_\_ method of the InstrumentBase class to inherit its properties and initialization.
* Initializes the following attributes: value (starting at 0), min\_value (0), max\_value (100), and label ('Value'). These define the range and label of the gauge.
* A QTimer object is created and its timeout signal is connected to the update\_value method. The timer is set to trigger every 1000 milliseconds (1 second), which will update the value of the gauge and cause it to redraw with the new value.

#### **update\_value Method**

* Increases the value attribute by 1. Then, it uses the modulo operation with (self.max\_value + 1) to ensure that the value stays within the range defined by min\_value and max\_value. This creates a cyclic behavior for the gauge value.
* Calls the update method, which is inherited from QWidget. This triggers the repainting of the gauge with the updated value.

#### **draw\_background Method**

* Calculates the center and radius of the gauge circle. The radius is determined as 0.8 times half of the smaller dimension (width or height) of the widget. The drawEllipse method is then used to draw the circular outline of the gauge.
* The font for drawing the scale values is set to bold Arial with size 10 and a light - gray color. A loop iterates from min\_value to max\_value with a step of 10. For each value, the corresponding angle on the circle is calculated using the value\_to\_angle method. Based on this angle, the position of the text is determined, and the scale value is drawn on the gauge using the drawText method.

#### **value\_to\_angle Method**

* This method calculates the angle on the gauge circle corresponding to a given value. It first finds the proportion of the value within the range ((value - min\_value) / (max\_value - min\_value)). This proportion is then multiplied by 2 \* math.pi to get the angle in radians within the full circle range. Finally, math.pi / 2 is subtracted to adjust the angle range to start from - math.pi / 2 and end at 3 \* math.pi / 2, which is suitable for the gauge's layout.

#### **draw\_indicator Method**

* Calculates the angle and length of the pointer based on the current value. The length of the pointer is a certain proportion (0.6) of the radius. The angle is obtained using the value\_to\_angle method. The drawLine method is used to draw the pointer with a red color and a width of 3 pixels.
* The font for displaying the current value and label is set to bold Arial with size 12 and a white color. The current value and label are drawn below the center of the gauge using the drawText method, making it easily visible to the user.

### **3. Thermometer Class (Thermometer)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the thermometer - specific attributes: value (20), min\_value (0), max\_value (100), and unit ('°C').
* A QTimer is created and connected to the update\_value method. The timer triggers every 2000 milliseconds (2 seconds) to update the temperature value.

#### **update\_value Method**

* Increments the value attribute by 5. Then, a modulo operation is used to keep the value within the min\_value to max\_value range, simulating the temperature change.
* Calls the update method to redraw the thermometer with the new temperature value.

#### **draw\_background Method**

* Uses the drawRect method to draw the rectangle representing the thermometer body. The rectangle's position (30, 50) and size (20, 200) are specified.
* The font for the scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the temperature range with a step of 10 calculates the y - coordinate for each scale value. The scale values are then drawn beside the thermometer using the drawText method.

#### **draw\_indicator Method**

* Calculates the y - coordinate of the temperature column based on the current value. The height of the column is determined by the proportion of the value within the min\_value to max\_value range. The drawRect method is used to draw the red temperature column.
* The font for displaying the temperature value and unit is set to bold Arial with size 12 and a white color. The current temperature value and unit are drawn near the top of the temperature column using the drawText method.

### **4. Manometer Class (Pressure Gauge)**

#### **\_\_init\_\_ Method**

* Calls the InstrumentBase class's \_\_init\_\_ method.
* Initializes the pressure gauge's attributes: value (1.0), min\_value (0), max\_value (2.0), and unit ('MPa').
* A QTimer is created and connected to the update\_value method. The timer triggers every 1500 milliseconds (1.5 seconds) to update the pressure value.

#### **update\_value Method**

* Increments the value by 0.1, rounds it using the round function, and then uses a modulo operation to keep the value within the min\_value to max\_value range, simulating pressure changes.
* Calls the update method to redraw the pressure gauge with the new pressure value.

#### **draw\_background Method**

* Gets the center of the pressure gauge and uses the drawArc method to draw the arc - shaped background. The arc's parameters include the size of the bounding rectangle (relative to the center, with width and height of 140) and the starting and spanning angles (45 \* 16 and 270 \* 16 respectively).
* The font for the pressure scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the pressure range with a step of 0.1 calculates the angle for each scale value. The scale values are then drawn on the pressure gauge using the drawText method.

#### **value\_to\_angle Method**

* Calculates the angle on the pressure gauge's arc corresponding to a given pressure value. It finds the proportion of the value within the range and multiplies it by 2.5 \* math.pi. Then, math.pi / 4 is subtracted to adjust the angle range from - math.pi / 4 to 5 \* math.pi / 4 for the arc - shaped layout.

#### **draw\_indicator Method**

* Calculates the angle and length of the pointer based on the current value. The pointer is drawn using the drawLine method with a blue color and a width of 3 pixels.
* The font for displaying the pressure value and label is set to bold Arial with size 12 and a white color. The current pressure value and label are drawn below the pressure gauge using the drawText method.

### **5. Voltmeter Class (Voltmeter)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the voltmeter's attributes: value (220), min\_value (0), max\_value (380), and unit ('V').
* A QTimer is created and connected to the update\_value method. The timer triggers every 3000 milliseconds (3 seconds) to update the voltage value.

#### **update\_value Method**

* Increments the value by 10 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating voltage changes.
* Calls the update method to redraw the voltmeter with the new voltage value.

#### **draw\_background Method**

* Uses the drawRect method to draw the rectangle representing the voltmeter body. The rectangle's position (30, 30) and size (100, 200) are specified.
* The font for the voltage scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the voltage range with a step of 20 calculates the y - coordinate for each scale value. The scale values are then drawn beside the voltmeter using the drawText method.

#### **draw\_indicator Method**

* Calculates the y - coordinate of the indicator bar based on the current value. The height of the bar is determined by the proportion of the value within the min\_value to max\_value range. The drawRect method is used to draw the green indicator bar.
* The font for displaying the voltage value and unit is set to bold Arial with size 12 and a white color. The current voltage value and unit are drawn near the top of the indicator bar using the drawText method.

### **6. Ammeter Class (Ammeter)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the ammeter's attributes: value (5), min\_value (0), max\_value (10), and unit ('A').
* A QTimer is created and connected to the update\_value method. The timer triggers every 2500 milliseconds (2.5 seconds) to update the current value.

#### **update\_value Method**

* Increments the value by 1 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating current changes.
* Calls the update method to redraw the ammeter with the new current value.

#### **draw\_background Method**

* Uses the drawRect method to draw the rectangle representing the ammeter body. The rectangle's position (30, 30) and size (100, 200) are specified.
* The font for the current scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the current range with a step of 1 calculates the y - coordinate for each scale value. The scale values are then drawn beside the ammeter using the drawText method.

#### **draw\_indicator Method**

* Calculates the y - coordinate of the indicator bar based on the current value. The height of the bar is determined by the proportion of the value within the min\_value to max\_value range. The drawRect method is used to draw the yellow indicator bar.
* The font for displaying the current value and unit is set to bold Arial with size 12 and a white color. The current current value and unit are drawn near the top of the indicator bar using the drawText method.

### **7. LevelMeter Class (Liquid Level Meter)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the liquid level meter's attributes: value (50), min\_value (0), max\_value (100), and unit ('%').
* A QTimer is created and connected to the update\_value method. The timer triggers every 2000 milliseconds (2 seconds) to update the liquid level value.

#### **update\_value Method**

* Increments the value by 5 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating liquid level changes.
* Calls the update method to redraw the liquid level meter with the new liquid level value.

#### **draw\_background Method**

* Uses the drawRect method to draw the rectangle representing the liquid level meter body. The rectangle's position (30, 30) and size (100, 200) are specified.
* The font for the liquid level scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the liquid level range with a step of 10 calculates the y - coordinate for each scale value. The scale values are then drawn beside the liquid level meter using the drawText method.

#### **draw\_indicator Method**

* Calculates the y - coordinate of the indicator bar based on the current value. The height of the bar is determined by the proportion of the value within the min\_value to max\_value range. The drawRect method is used to draw the orange indicator bar.
* The font for displaying the liquid level value and unit is set to bold Arial with size 12 and a white color. The current liquid level value and unit are drawn near the top of the indicator bar using the drawText method.

### **8. FlowMeter Class (Flow Meter)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the flow meter's attributes: value (100), min\_value (0), max\_value (200), and unit ('L/min').
* A QTimer is created and connected to the update\_value method. The timer triggers every 1000 milliseconds (1 second) to update the flow value.

#### **update\_value Method**

* Increments the value by 10 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating flow changes.
* Calls the update method to redraw the flow meter with the new flow value.

#### **draw\_background Method**

* Calculates the center of the flow meter and uses the drawEllipse method to draw the circular outline with a radius of 60.
* The font for the flow scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the flow range with a step of 20 calculates the angle for each scale value using the value\_to\_angle method. The scale values are then drawn on the flow meter using the drawText method.

#### **value\_to\_angle Method**

* Calculates the angle on the flow meter's circle corresponding to a given flow value. It finds the proportion of the value within the range and multiplies it by 2 \* math.pi. Then, math.pi / 2 is subtracted to adjust the angle range from - math.pi / 2 to 3 \* math.pi / 2 for the circular layout.

#### **draw\_indicator Method**

* Calculates the angle and length of the pointer based on the current value. The pointer is drawn using the drawLine method with a magenta color and a width of 3 pixels.
* The font for displaying the flow value and label is set to bold Arial with size 12 and a white color. The current flow value and label are drawn below the flow meter using the drawText method.

### **9. Tachometer Class (Tachometer)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the tachometer's attributes: value (1000), min\_value (0), max\_value (2000), and unit ('RPM').
* A QTimer is created and connected to the update\_value method. The timer triggers every 1500 milliseconds (1.5 seconds) to update the rpm value.

#### **update\_value Method**

* Increments the value by 100 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating rpm changes.
* Calls the update method to redraw the tachometer with the new rpm value.

#### **draw\_background Method**

* Calculates the center of the tachometer and uses the drawEllipse method to draw the circular outline with an adjusted size.
* The font for the rpm scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the rpm range with a step of 200 calculates the angle for each scale value using the value\_to\_angle method. The scale values are then drawn on the tachometer using the drawText method.

#### **value\_to\_angle Method**

* Calculates the angle on the tachometer's circle corresponding to a given rpm

https://www.mojie.cyou/#/dashboard

不完整

### **9. Tachometer Class (Tachometer)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the tachometer's attributes: value (1000), min\_value (0), max\_value (2000), and unit ('RPM').
* A QTimer is created and connected to the update\_value method. The timer triggers every 1500 milliseconds (1.5 seconds) to update the rpm value.

#### **update\_value Method**

* Increments the value by 100 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating rpm changes.
* Calls the update method to redraw the tachometer with the new rpm value.

#### **draw\_background Method**

* Calculates the center of the tachometer and uses the drawEllipse method to draw the circular outline with an adjusted size.
* The font for the rpm scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the rpm range with a step of 200 calculates the angle for each scale value using the value\_to\_angle method. The scale values are then drawn on the tachometer using the drawText method.

#### **value\_to\_angle Method**

* Calculates the angle on the tachometer's circle corresponding to a given rpm value. It determines the proportion of the rpm value within the range ((value - min\_value) / (max\_value - min\_value)). This proportion is multiplied by 2 \* math.pi to obtain the angle in radians within the full - circle range. Subsequently, math.pi / 2 is subtracted to adjust the angle range to start from - math.pi / 2 and end at 3 \* math.pi / 2, which is suitable for the tachometer's circular layout.

#### **draw\_indicator Method**

* Calculates the angle and length of the pointer based on the current value. The pointer length is determined according to the circle's radius and the position is calculated using the angle obtained from the value\_to\_angle method. The drawLine method is used to draw the pointer with a cyan color and a width of 3 pixels.
* The font for displaying the rpm value and label is set to bold Arial with size 12 and a white color. The current rpm value and label are drawn below the tachometer using the drawText method, making it easy for the user to read the information.

### **10. Hygrometer Class (Hygrometer)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the hygrometer's attributes: value (50), min\_value (0), max\_value (100), and unit ('%RH').
* A QTimer is created and connected to the update\_value method. The timer triggers every 2500 milliseconds (2.5 seconds) to update the humidity value.

#### **update\_value Method**

* Increments the value by 5 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating humidity changes.
* Calls the update method to redraw the hygrometer with the new humidity value.

#### **draw\_background Method**

* Uses the drawRect method to draw the rectangle representing the hygrometer body with adjusted position and size.
* The font for the humidity scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the humidity range with a step of 10 calculates the y - coordinate for each scale value. The scale values are then drawn beside the hygrometer using the drawText method.

#### **draw\_indicator Method**

* Calculates the y - coordinate of the indicator bar based on the current value. The height of the bar is determined by the proportion of the value within the min\_value to max\_value range. The drawRect method is used to draw the purple indicator bar.
* The font for displaying the humidity value and unit is set to bold Arial with size 12 and a white color. The current humidity value and unit are drawn near the top of the indicator bar using the drawText method.

### **11. PowerMeter Class (Power Meter)**

#### **\_\_init\_\_ Method**

* Inherits from the InstrumentBase class's \_\_init\_\_ method.
* Initializes the power meter's attributes: value (1000), min\_value (0), max\_value (2000), and unit ('W').
* A QTimer is created and connected to the update\_value method. The timer triggers every 2000 milliseconds (2 seconds) to update the power value.

#### **update\_value Method**

* Increments the value by 100 and uses a modulo operation to keep the value within the min\_value to max\_value range, simulating power changes.
* Calls the update method to redraw the power meter with the new power value.

#### **draw\_background Method**

* Calculates the center of the power meter and uses the drawEllipse method to draw the circular outline with an adjusted size.
* The font for the power scale values is set to bold Arial with size 10 and a light - gray color. A loop based on the power range with a step of 200 calculates the angle for each scale value using the value\_to\_angle method. The scale values are then drawn on the power meter using the drawText method.

#### **value\_to\_angle Method**

* Calculates the angle on the power meter's circle corresponding to a given power value. It finds the proportion of the power value within the range ((value - min\_value) / (max\_value - min\_value)). This proportion is multiplied by 2 \* math.pi to obtain the angle in radians within the full - circle range. Then, math.pi / 2 is subtracted to adjust the angle range to start from - math.pi / 2 and end at 3 \* math.pi / 2, which is appropriate for the power meter's circular layout.

#### **draw\_indicator Method**

* Calculates the angle and length of the pointer based on the current value. The pointer length is calculated based on the circle's radius, and the position is determined using the angle from the value\_to\_angle method. The drawLine method is used to draw the pointer with a dark yellow color and a width of 3 pixels.
* The font for displaying the power value and label is set to bold Arial with size 12 and a white color. The current power value and label are drawn below the power meter using the drawText method, allowing the user to easily read the power information.

### **Main Program Section**

* In the main part of the program, an instance of QApplication named app is created. This is the entry point of the Qt application and is responsible for managing the application's life cycle and event loop.
* A QGridLayout object named layout is created. The grid layout is used to arrange multiple instruments in a regular grid format on the interface.
* Ten different instrument instances are created and added to the widgets list. Each instrument instance has its own unique attributes and the ability to update its value dynamically through timers.
* Through a loop, each instrument is added to the layout. The column index col is incremented for each added instrument. When col reaches 5, a new row is started, and col is reset to 0 while the row index row is incremented. This ensures that the instruments are arranged in a 5 - column grid on the interface.
* A main widget main\_widget is created, and the layout is set for it. This allows the instruments in the layout to be displayed within the main window. Then, the show method is called to display the main window, making the instrument interface visible to the user.
* Finally, app.exec\_() is called to start the application's event loop. In the event loop, the application continuously processes various events such as user input and timer triggers. When the user closes the window, the program exits the event loop and terminates the application.